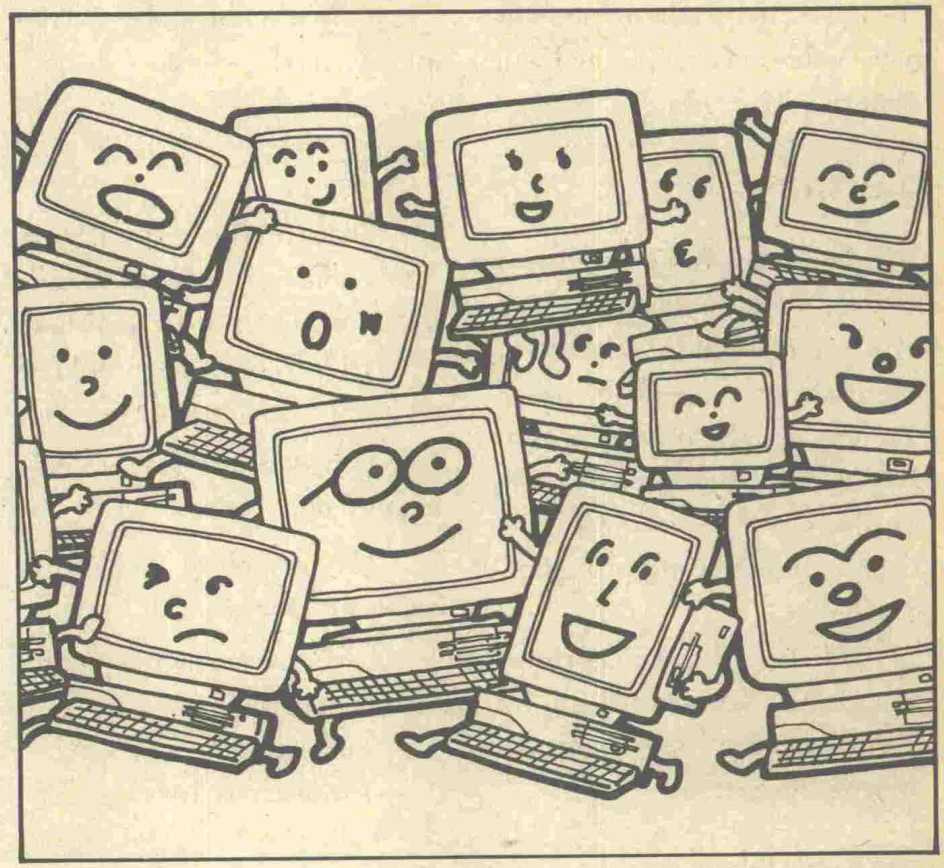
บุญเลิศ เอี่ยมทัศนา

โปรแกรมนักวาด
ตอนที่ 1 : เริ่มต้น

กล่าวถึงวิธีการเขียนโปรแกรมวาดรูปและเรื่องของการเขียนโปรแกรมบนวินโดวส์



หตุหนึ่งที่ผมขอพักเรื่องเพลซเอเบิล-เมตาไฟล์ไว้ก่อน ก็เพื่อจะนำการเขียนโปรแกรมวาดรูปดังเช่นเพนต์บรัชมากล่าวถึง ซึ่งผมจะถือโอกาสนี้ต้อนรับผู้ที่เริ่มสนใจเขียนโปรแกรมบนวินโดวส์ด้วย เพื่อให้สมกับเป็นฉบับที่ 101 ยินดีต้อนรับครับ

สองภาษา

และเพื่อให้ทั้งผู้ที่สนใจภาษาปาสคาลหรือซีได้เริ่มไปพร้อมกัน ผมจะกล่าวทั้งสองภาษานี้คู่กัน ด้วยการเขียนโปรแกรมแบบโอโอพี คอมพายล์เลอร์ที่คุณจะเลือกใช้ได้คือ

เทอร์โบปาสคาล

ได้แก่

- TPW คือ Turbo Pascal for Windows 1.5 หรือ 1.0 ซึ่งต้องใช้กับวินโดวส์ 3.0
- BPW คือที่อยู่ในชุด Borland Pascal with Objects 7.0

หากจะเลือกในระหว่างสองชุดนี้ ควรใช้ TPW 1.5 เพราะใช้เนื้อที่น้อยกว่ากันมาก (ประมาณ 13 ต่อ 30 เมกะไบต์) ผมใช้ TPW 1.5 เขียนโปรแกรมแล้วคอมพายล์ด้วย BPW 7.0 ซ้ำอีกครั้งหนึ่ง

เทอร์โบซี

ผมคุ้นเคยกับ C++ ของบริษัทบอร์แลนด์น้อยมาก เพราะไม่ได้ติดตามมาตลอด โดยที่ในการเขียนบทความชุดนี้ผมใช้ TCW 3.0 และจะคอมพายล์ตรวจสอบในบางโปรแกรมเมื่อสงสัยกับ BCW 3.1 จึงคิดว่าน่าจะใช้ได้กับ

- TCW คือ Turbo C++ for Windows ทั้งรุ่น 3.0 และ 3.1
- BPW คือ Borland C++ for Windows ทั้งรุ่น 3.0 และ 3.1

หากจะเลือกในสองชุดนี้ ผมคิดว่าควรใช้ TCW มากกว่าด้วยเหตุผลเดียวกันคือใช้เนื้อที่ในดิสค์น้อยกว่ากัน (ประมาณ 20 ต่อ 50 เมกะไบต์)

และถ้าคุณจะเลือกระหว่างสองภาษานี้ผมเห็นว่าควรใช้เทอร์โบปาสคาล

เพราะจะทำให้เข้าถึงการเขียนโปรแกรมบนวินโดวส์ได้เร็วกว่า เนื่องจากเทอร์โบปาสคาลไม่มีเรื่องจุกจิก เมื่อเข้าใจวิธีแล้วจะเขียนด้วยภาษาใดเป็นเรื่องง่าย และผมไม่เห็นข้อแตกต่างในผลการรันไม่ว่าโปรแกรมนั้นจะเขียนด้วยซีธรรมดา หรือ C++ หรือเทอร์โบปาสคาล เพราะต่างก็เรียกใช้ฟังก์ชันของวินโดวส์เหมือนกันและด้วยวิธีเดียวกัน

ที่ว่าเริ่มต้น

ที่ผมขอต้อนรับผู้เริ่มต้น เป็นการเริ่มต้นของการเขียนโปรแกรมบนวินโดวส์ครับ จะต้องเป็นผู้ที่เขียนโปรแกรมในภาษาปาสคาลหรือซีเป็นแล้ว ใช้วินโดวส์เป็น และตามคู่มือของทั้งสองภาษาบ่งว่าต้องเขียนโปรแกรมแบบโอโอพีเป็น คือตามที่ผมกล่าวไว้ในหนังสือ "การโปรแกรมแบบโอโอพีด้วยเทอร์โบปาสคาล"

แต่ในหนังสือนี้หนักไปในทางออกแบบโปรแกรมจึงยังไม่จำเป็น คิดว่าตามที่ผมจะนำมากล่าวสอดแทรกในแต่ละตอนก็เพียงพอ

การเขียนโปรแกรมวินโดวส์ 1

การเขียนโปรแกรมบนวินโดวส์คือการเขียนโปรแกรมเพื่อใช้สิ่งอำนวยความสะดวกที่วินโดวส์จัดไว้ให้ ซึ่งสามารถแบ่งออกได้เป็น 3 ระดับหรือ 3 เรื่องคือ

1. พื้นฐาน

เป็นเรื่องที่ผู้เขียนโปรแกรมบนวินโดวส์ทุกคนจะต้องทราบ เพื่อใช้ประกอบในการเขียนโปรแกรมในเรื่องอื่นต่อไป ได้แก่

- เรื่องที่ต้องทราบก่อนคือ หน้าต่าง เมนู ไดอะล็อกบ็อกซ์ และคอนโทรล
- เรื่องที่อาจทราบภายหลังคือ ไอคอน เคอร์เซอร์ ไทเมอร์ สตริง และคีย์ด่วน
- เรื่องที่ควรทราบคือ การจัดหน่วยความจำ

2. ทั่วไป

เป็นเรื่องที่ใช้ในการเขียนโปรแกรมโดยทั่วไปคือ การแสดงผล หมายถึงทั้งภาพและข้อความ เพราะบนวินโดวส์ทุกอย่างเป็นกราฟิก และกับอุปกรณ์ทุกชนิดกับการจัดเก็บข้อมูล คือ

- การแสดงภาพ ซึ่งหมายรวมถึงบิตแม็พด้วย
- การแสดงข้อความ ซึ่งจะเป็นเรื่องของเท็กซ์และข้อความ
- คีย์บอร์ดและเม้าส์ เพื่อสนับสนุนการแสดงผลทางจอภาพ
- เครื่องพิมพ์
- ไฟล์และเมตาไฟล์

3. เรื่องเฉพาะ

มีในบางโปรแกรมคือบางแอพพลิเคชันเท่านั้น คือการส่งผ่านข้อมูลระหว่างแอพพลิเคชัน ได้แก่

- Clipboard
- Dynamic Data Exchange
- Object Linking and Embedding

ซึ่งในการนำมากล่าวไม่จำเป็นต้องกล่าวเรียงลำดับจากพื้นฐานถึงเรื่องเฉพาะ โดยเฉพาะเรื่องพื้นฐานกับเรื่องทั่วไปสามารถนำมากล่าวรวมกันได้

การเขียนโปรแกรมบนวินโดวส์ 2

การเขียนโปรแกรมบนวินโดวส์เป็นการเขียนโปรแกรมจากเหตุที่เกิดขึ้นจากผู้ใช้ (event driver) ซึ่งวินโดวส์จะส่งเมสเสจ (message) จากเหตุที่เกิดขึ้น เช่น ผู้ใช้คลิ๊กเม้าส์ในหน้าต่างหรือเลือกรายการในเมนู เป็นต้น ให้เราดำเนินการกับเหตุนั้น ๆ

การเขียนโปรแกรมบนวินโดวส์ 3

การเขียนโปรแกรมบนวินโดวส์โดยเฉพาะเมื่อใช้เทอร์โบทั้งสอง (ปาสคาลหรือซี) เขียนได้ 2 วิธีคือ

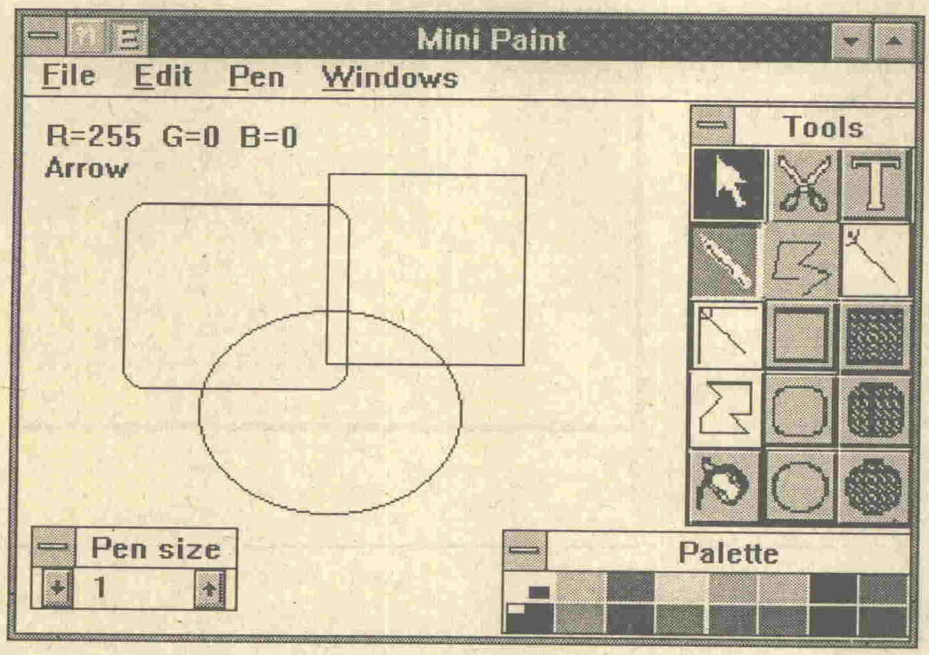
- เขียนตามแบบ (ดั้งเดิมที่เคยเขียนกันมาและยังคงใช้ได้ต่อไป)
- เขียนแบบโอโอพี (Object-Oriented programming)

บทความนี้จะใช้วิธีเขียนโปรแกรมแบบโอโอพี เพราะจะทำให้เข้าถึงการเขียนโปรแกรมบนวินโดวส์ได้เร็วกว่าและง่ายกว่าการเขียนโปรแกรมตามแบบ

โปรแกรมที่จะเขียน

เพื่อให้เห็นภาพคร่าว ๆ ว่าจะดำเนินการอย่างไรต่อไป ขอนำโปรแกรมที่จะเขียนมากล่าวถึงโดยสังเขปคือ จะเขียนเป็นโปรแกรมอย่างเพนต์บรัช ซึ่งเมื่อรันจะให้ผลการรันดังรูปที่ 1 ซึ่งผมจะต่อเติมโปรแกรมในแต่ละตอนจนสมบูรณ์

จะทำให้ได้กล่าวถึง การเขียนโปรแกรมบนวินโดวส์ตามที่กล่าวมามากกว่าครึ่ง



รูปที่ 1 ผลการรันของโปรแกรมที่จะเขียน

เริ่มที่ปาสคาล

ไม่ใช่ผมไม่ให้เกียรติซี แต่เป็นเพราะของปาสคาลมีเรื่องที่กล่าวถึงน้อยกว่า และสำหรับท่านที่จะใช้เทอร์โบซีขอให้อ่านด้วย

ผมจะให้ทดลองป้อน คอมพายล์ และรันโปรแกรมคือ หลังจากที่รัน TPW หรือ BPW แล้ว ให้ดำเนินการดังนี้

1. เปิดไฟล์ด้วยชื่อ HELLO.PAS ความหมายของผมคือให้เปิดไฟล์ด้วย File/Open (ในการนี้ขอให้เตรียมไดเรคทอรีไว้ก่อนครับ) เพราะเป็นไฟล์ใหม่ให้ป้อนชื่อไฟล์เป็น HELLO.PAS (หากเป็นไฟล์เก่า ให้เลือกจากรายการ)

2. ป้อนโปรแกรมตามลิสติงที่ 1P (เมื่อป้อนเสร็จคุณอาจรันได้ทันที แต่ไม่ใช่วิธีที่ถูกต้อง)

3. คอมพายล์ด้วยรายการ compile/Compile เมื่อไม่มีที่ผิดให้เก็บลงไฟล์ด้วยรายการ File/Save

4. รันด้วยรายการ Run/Run จะได้ผลการรันดังรูปที่ 2

ลิสติงที่ 1P

```
program Hello;

uses WObjects;

type
  TMyApplication = object(TApplication)
    procedure InitMainWindow; virtual;
  end;


procedure TMyApplication.InitMainWindow;
begin
  MainWindow := New(PWindow, Init(nil, 'Hello World!'));
end;


var
  MyApp: TMyApplication;

begin
  MyApp.Init('MyProgram');
  MyApp.Run;
  MyApp.Done;
end.
```

ตามที่ผมให้เลือกการดำเนินการจากเมนู อาจใช้คีย์ด่วนที่ให้ไว้ท้ายรายการหรือเลือกจากปุ่มต่าง ๆ ที่ speed bar (อยู่ใต้เมนูบาร์) ได้เช่นกัน

เมื่อคอมพายล์จะได้ไฟล์ HELLO.EXE ซึ่งโปรแกรมตัวอย่างเช่นนี้ไม่จำเป็นต้องเก็บไฟล์ .EXE ไว้ หากต้องการรันใหม่ให้เปิดไฟล์ .PAS มารันด้วย Run/Run

ตามด้วยซี

เพื่อทดลองป้อน คอมพายล์ และรันเช่นกัน ซึ่งสำหรับเทอร์โบซีสามารถดำเนินได้ทั้งเปิดโปรเจ็กต์ก่อนหรือป้อนโปรแกรมก่อน จะกล่าวตามวิธีป้อนโปรแกรมก่อนคือ

1. เปิดไฟล์ด้วยชื่อ HELLO.CPP ป้อนโปรแกรมตามลิสติงที่ 1C แล้วเก็บลงไฟล์

2. เปิดโปรเจ็กต์ ตามที่จะกล่าว

ในหัวข้อ โปรเจ็กต์ของเทอร์โบซี

3. คอมพายล์ด้วย Compile/Make ซึ่งจะเป็นการคอมพายล์และลิงค์

4. รันด้วย Run/Run (ซึ่งอาจไม่คอมพายล์ก่อนได้) จะได้ผลการรันดังรูปที่ 2 เช่นกัน

ซึ่งเมื่อดำเนินการตามนี้จะได้ไฟล์ต่าง ๆ มากมาย รวมทั้งไฟล์ .EXE ด้วย ไม่จำเป็นต้องเก็บไว้คงเก็บไว้แต่ไฟล์ .CPP และ .PRJ หากต้องการรันใหม่ให้เปิดโปรเจ็กต์ แล้วเลือกไฟล์ .PRJ ที่ต้องการ และรันด้วย Run/Run (จะเป็นการคอมพายล์และรันใหม่ได้ไฟล์ที่ไม่ต้องการเก็บเช่นเดิม)

โปรเจ็กต์ของเทอร์โบซี

โปรเจ็กต์ไม่ได้มีความหมายใดเป็นพิเศษ เมื่อต้องการรวมสิ่งต่าง ๆ เข้าด้วยกัน บอร์แลนด์จะให้ทำเป็นโปรเจ็กต์ สำหรับกรณีนี้เมื่อจะรวมไฟล์ .CPP และอย่างน้อยที่สุดคือไฟล์ .DEF และต่อไปจะมีไฟล์ .RES และอื่น ๆ อีก จึงให้ทำเป็นโปรเจ็กต์และเป็นไฟล์ .PRJ ซึ่งจะดำเนินการได้ดังนี้

เตรียมการ

ขอให้เปิดไฟล์ด้วยชื่อ STANDARD.DEF ในไดเรคทอรีเดียวกับที่เก็บไฟล์ ป้อนข้อความตามรูปที่ 3 เก็บลงไฟล์แล้วปิดหน้าต่างนี้

หรืออาจจะลอกมาจากไฟล์ OWL.DEF ในไดเรคทอรี C:\TCWIN\OWL\LIB หรือ C:\BORLANDC\OWL\LIB

หรือไม่ใช้ไฟล์ STANDARD.DEF แต่เมื่อให้เลือกไฟล์นี้ ให้ป้อนชื่อไฟล์เป็น OWL.DEF รวมทั้งไดรฟ์และไดเรคทอรีด้วย

รูปที่ 2
ผลการรันของโปรแกรม Hello

ลิสติงที่ 1C

```
// Hello

#include <owl.h>

class THelloApp : public TApplication
{
public:
 THelloApp(LPSTR AName, HINSTANCE hInstance, HINSTANCE hPrevInstance,
  LPSTR lpCmdLine, int nCmdShow)
  : TApplication(AName, hInstance, hPrevInstance, lpCmdLine, nCmdShow) {};
 virtual void InitMainWindow();
};

void THelloApp::InitMainWindow()
{
 MainWindow = new TWindow(NULL, "Hello World!");
}

int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
 LPSTR lpCmdLine, int nCmdShow)
{
 THelloApp HelloApp ("HelloApp", hInstance, hPrevInstance,
  lpCmdLine, nCmdShow);
 HelloApp.Run();
 return HelloApp.Status;
}
```

```
EXETYPE WINDOWS
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE MULTIPLE
HEAPSIZE 4096
STACKSIZE 5120
```

รูปที่ 3
ข้อความในไฟล์ STANDARD.DEF

เปิดโปรเจ็กต์

ในการเปิดโปรเจ็กต์ผมได้ดำเนินการตามคู่มือแล้วไม่สำเร็จ และผมยังไม่มีเวลาจะให้ จึงขอนำวิธีที่ผมใช้มากล่าวถึง ขอให้ดำเนินการดังนี้คือ

1. เปิดโปรเจ็กต์เก่า คือเลือกรายการ Project \ Open project จะแสดงไฟล์ไดอะล็อกบ็อกซ์ให้เลือกไฟล์โปรเจ็กต์เก่า เช่น

```
C: \ BORLANDC \ OWL \ EXAMPLES \ helloapp.prj
```

2. เปิดโปรเจ็กต์ใหม่ ขณะนี้ยังเปิดโปรเจ็กต์เก่าค้างอยู่ ให้เปิดโปรเจ็กต์ใหม่ในไดเรคทอรีที่ต้องการด้วยการป้อนชื่อไฟล์ใหม่ ซึ่งควรให้ชื่อเดียวกับโปรแกรมเช่น HELLO.PRJ

3. เลือกรายการ Project/Add item จะแสดงไดอะล็อกบ็อกซ์ให้ป้อนชื่อไฟล์ ให้ป้อนหรือเลือกไฟล์ .CPP ที่ต้องการ แล้วเลือกปุ่ม Add

4. จะแสดงไดอะล็อกบ็อกซ์ซ้ำเดิมอีกให้เลือกไฟล์ STANDARD. DEF ตามที่เตรียมการไว้แล้วเลือกปุ่ม Done

เป็นการเปิดโปรเจ็กต์สำหรับใช้กับโปรแกรม *.CPP ซึ่งอาจป้อนไว้ก่อนหรือป้อนภายหลังเปิดโปรเจ็กต์

ออปเจ็กต์กับสกุล

ก่อนที่จะเดินเรื่องต่อไปขอนำเรื่องของโอโอพีมาแทรกก่อนคือ

- การเขียนโปรแกรมแบบโอโอพีเป็นการเขียนโปรแกรมโดยมุ่งที่ออปเจ็กต์

- ออปเจ็กต์คือสิ่งใดก็ได้ที่สามารถทำเป็นแบบจำลองได้

- ทำเป็นแบบจำลองได้ด้วยการสร้างเป็นสกุล (class) มีลักษณะคล้ายเรคอร์ด (RECORD ในภาษาปาสคาล หรือ struct ในภาษาซี) ประกอบด้วยฟิลด์ข้อมูลและฟิลด์กิจกรรม (method) คือส่วนหัวของโปรแกรมย่อย

- นำไปใช้โดยกำหนดตัวแปรของสกุลเรียกว่าอินสแตนซ์ (instance) แล้วให้กำเนิดเป็นออปเจ็กต์ตามที่จะกล่าวต่อไป

เรื่องของโอโอพียังมีอีกตามที่ผมจะแทรกต่อ ๆ ไป ที่นำส่วนนี้มากล่าวก่อนเพราะว่า ตัวโปรแกรมของวินโดวส์เองก็เขียนในแบบโอโอพี ถึงแม้จะไม่ใช้ภาษาที่เป็นโอโอแอล จึงมีการนำคำทางด้านโอโอพีมาใช้ ๆ เช่น ออปเจ็กต์และอินสแตนซ์ ซึ่งควรทราบไว้จะได้ไม่สับสน

ภาษาที่เป็นโอโอแอล (Object-Oriented Language) คือ ภาษาที่ออกแบบมาเพื่อใช้ในการเขียนโปรแกรมแบบโอโอพี ดังเช่นเทอร์โบปาสคาลและเทอร์โบซีตามที่กล่าวมา และเพราะวินโดวส์ไม่ได้เขียนด้วยภาษาที่เป็นโอโอแอลจึงทำให้ไม่เห็นสกุลและสืบสกุลไม่ได้

โปรแกรมที่สอง

ผมจะยังไม่อธิบายโปรแกรมแรก ขอให้ทดลองรัน (ป้อน คอมพายล์ และรัน) โปรแกรม Cursor AndIcon ตามลิสติงที่ 2P หรือ 2C โดยเก็บไว้ในไฟล์ CURICO.PAS หรือ CURICO.CPP

เมื่อรันโปรแกรมนี้จะได้เคอร์เซอร์ (ของเม้าส์) เป็นรูปตัว I เมื่อคลิ๊กเม้าส์ซ้ายจะได้ผลการรันตามรูปที่ 4 และเมื่อมินิไมส์จะได้ไอคอนรูปอักษร i มีเรื่องที่จะกล่าวถึงหลายเรื่องตามที่จะกล่าวในแต่ละหัวข้อต่อไป

แอพพลิเคชัน

แอพพลิเคชันคือโปรแกรมที่สมบูรณ์แล้วทั้งโปรแกรม ส่วนโปรแกรมอาจจะเป็นทั้งหมดหรือเป็นเพียงส่วนหนึ่งของโปรแกรม ผมจะใช้ควบคู่กันไปตามที่เห็นสมควร

โปรแกรมบนวินโดวส์

อาจแบ่งโปรแกรมบนวินโดวส์หรือที่เรียกว่าวินโดวส์แอพพลิเคชันออกได้เป็น 2 ส่วนคือ โปรแกรมหลักกับโปรแกรมย่อย โดยที่โปรแกรมย่อยคือโปรแกรมของหน้าต่าง จึงแบ่งเป็นโปรแกรมหลักกับโปรแกรมของหน้าต่าง

โปรแกรมหลัก

เมื่อเขียนโปรแกรมตามแบบด้วยภาษาซีจะต้องใช้ชื่อว่า WinMain ทำหน้าที่

1. ขึ้นทะเบียนคลาส (class) ของหน้าต่าง

2. ให้กำเนิดหน้าต่าง (หน้าต่างเป็นออปเจ็กต์)

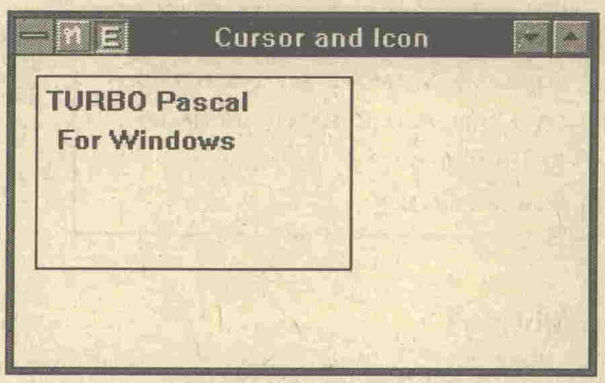
3. ให้แสดงหน้าต่าง (เมื่อให้กำเนิดหน้าต่างแล้วจะต้องให้แสดงหรือ

ไม่แสดง)

4. เข้าทำงานเป็นวงรอบซึ่งจะขอเรียกว่าวงรอบการรัน ดำเนินการในเรื่อง

- ถอนเมสเสจจากคิว (วินโดวส์จะส่งเมสเสจให้แก่แอพพลิเคชันต่าง ๆ โดยใส่ไว้ในคิวของแอพพลิเคชันนั้น)
- ส่งให้วินโดวส์ตีความเมสเสจ
- ส่งให้วินโดวส์ส่งต่อให้แก่โปรแกรมของหน้าต่าง (จะเห็นว่าโปรแกรมหลักกับโปรแกรมของหน้าต่างไม่ได้ติดต่อกันโดยตรง)

5. ทำงานอยู่ในวงรอบการรันจนกว่าผู้ใช้จะยุติการรัน



รูปที่ 4
ผลการรันของโปรแกรม CursorAndIcon

โปรแกรมของหน้าต่าง

คำนี้เป็นพหูพจน์คือ ถ้ามีหน้าต่างหลายบานก็จะต้องมีโปรแกรมหน้าต่างแต่ละบาน แต่หน้าต่างบางบานอาจใช้โปรแกรมเดียวกันได้ เมื่อมีหน้าต่างบานเดียวหรือหลายบานก็ตามจะต้องมีบานหนึ่งเป็นหน้าต่างหลัก ซึ่งเมื่อเขียนโปรแกรมตามแบบด้วยซีนิยมใช้ชื่อฟังก์ชันนี้ว่า WndProc

โปรแกรมของหน้าต่างคือโปรแกรมงานของแอพพลิเคชันนั้น ดังเช่นในโปรแกรม Cursor AndIcon (ขอให้ดู

ลิสติงที่ 2C

```
// CursorAndIcon

#include "owl.h"

#define classnme "1ci"
#define appname "curicon"
#define thetitle "Cursor and Icon"

_CLASSDEF (WndProC)
class  WndProC : public TWindow {
public:
 HDC wndDC;
 BOOL pendwn;
 WndProC(PTWindowsObject AParent, LPSTR ATitle);
 virtual LPSTR GetClassName();
 virtual void GetWindowClass(WNDCLASS& AWndClass);
 virtual void WMLButtonDown(RTMessage) =
                [WM_FIRST + WM_LBUTTONDOWN];
 void  DemoDisp(HDC adc);
};

/* ---------- WndProC  routine ---------- */

WndProC::WndProC(PTWindowsObject AParent, LPSTR ATitle)
 : TWindow(AParent, ATitle)
{
}


LPSTR WndProC::GetClassName()
{
  return classnme;
}


void WndProC::GetWindowClass(WNDCLASS& AWndClass)
{
 TWindow::GetWindowClass(AWndClass);
 AWndClass.hIcon = LoadIcon(0, IDI_ASTERISK);
 AWndClass.hCursor = LoadCursor(0, IDC_IBEAM);
}


void  WndProC::WMLButtonDown(RTMessage)
{
 HDC wndDC;
```

ลิสติงที่ 2C (ต่อ)

```
 wndDC = GetDC(HWindow);
 DemoDisp(wndDC);
 ReleaseDC(HWindow, wndDC);
}

void  WndProC::DemoDisp(HDC adc)
{
 char txt1[] = "For Windows";

 Rectangle(adc, 10,10,160,110);
 TextOut(adc, 15,15, "TURBO C++",9);
 TextOut(adc, 20,35, txt1, lstrlen(txt1));
}

/* ---------------- TheAppC Class ---------------- */

class TheAppC : public TApplication {
public:
 TheAppC (LPSTR name, HINSTANCE hInst, HINSTANCE hPrvInst,
          LPSTR lpCmd, int nCmdShow)
   : TApplication (name, hInst, hPrvInst, lpCmd, nCmdShow) {};

virtual void InitMainWindow();
};

void TheAppC::InitMainWindow()
{
 MainWindow = new WndProC (NULL, thetitle);
}

/* ---------------- MAIN PROGRAM ---------------- */

int PASCAL WinMain(HINSTANCE hInst, HINSTANCE hPrvInst,
                   LPSTR lpCmd, int nCmdShow)
{
  TheAppC app (appname, hInst, hPrvInst,
                   lpCmd, nCmdShow);
  app.Run();
  return (app.Status);
}
```

ในรูปผลการรันก่อน) งานคือการแสดงภาพเป็นรูปสี่เหลี่ยมและแสดงข้อความ 2 ข้อความ

เมื่อเป็นโอโอพี

โดยที่การเขียนโปรแกรมบนวินโดวส์จะมีส่วนที่ซ้ำ ๆ กันในทุกแอพพลิเคชัน แตกต่างกันในรายละเอียดเล็กน้อย โดยเฉพาะในส่วนของโปรแกรมหลัก ในเทอร์โบทั้งคู่ต้องการให้เราเริ่มต้นเขียนโปรแกรมงานได้ทันที (คือไม่ต้องเขียนโปรแกรมหลัก) โดยให้เขียนแบบโอโอพี จึงได้นำโปรแกรมทั้งสองส่วนมาทำเป็นสกุลคือ

- นำโปรแกรมหลักมาทำเป็นสกุล TApplication ในความหมายว่าเป็นสกุลของแอพพลิเคชัน เพราะในวินโดวส์เมื่อกล่าวถึงแอพพลิเคชันจะหมายถึงโปรแกรมหลักเป็นหลัก

- นำโปรแกรมของหน้าต่างในส่วนที่ยังไม่เกี่ยวกับโปรแกรมงานมาทำเป็นสกุล TWindow

เพื่อให้เราสร้างผู้สืบสกุลเพื่อเขียนโปรแกรมต่อและตามที่จะกล่าวต่อไป นอกจากนี้ยังมีสกุลอื่นอีก เช่น สกุลของไดอะล็อกบ็อกซ์ รวมเรียกว่าออปเจ็กต์วินโดวส์ (ObjectWindow)

โอโอพีต่อ

วัตถุประสงค์ของการเขียนโปรแกรมแบบโอโอพีคือให้สามารถนำโปรแกรมที่เขียนไว้มาใช้ได้อีก โดยให้ทำเป็นสกุลเพื่อให้สามารถปรับปรุงคือทั้งเพิ่มและเปลี่ยนแปลงการทำงานของกิจกรรมเดิม โดยไม่ต้องแก้ในซอร์สโค้ดเดิมคือในไฟล์ .PAS หรือ .CPP

ตามที่จะนำมากล่าวในหัวข้อนี้ โดยจะขอยกโปรแกรม CursorAndIcon ในส่วนของการกำหนดสกุล TheAppC ที่อยู่ตอนท้ายมาเป็นตัวอย่าง คือจะกล่าวถึงทางด้านโอโอพีแต่เพียงอย่างเดียวก่อน และจะถือภาษาปาสคาลเป็นหลัก ขอให้ดูในลิสติงที่ 2P ครับ

จุดประสงค์ของการสร้างสกุลนี้ (ทางด้านโอโอพี) คือต้องการเปลี่ยนแปลงการทำงานของกิจกรรม InitMainWindow ใน TApplication เดิม ให้ใช้หน้าต่างของเราคือ WndProC และให้ชื่อหน้าต่าง จึงได้ดำเนินการดังนี้

การสร้างสกุล

แบ่งออกเป็น 2 ส่วนคือ การกำหนดสกุลกับการเขียนโปรแกรมของ

กิจกรรม กล่าวคือ

1. **การกำหนดสกุล** ทั้งสองภาษาให้กำหนดสกุลโดยทำเป็นแบบข้อมูลมีลักษณะคล้ายเรคอร์ดในภาษาปาสคาลหรือสตรัคในภาษาซี ดังการกำหนดสกุล TheAppC คือ

- ใช้คำสงวน OBJECT หรือ class
- ให้สืบสกุลจาก TApplication
- ภายในประกอบด้วยฟิลด์ข้อมูลและฟิลด์กิจกรรม แต่ตามตัวอย่างนี้มีฟิลด์กิจกรรมเพียงฟิลด์เดียวคือ InitMainWindow

2. **โปรแกรมของกิจกรรม** เมื่อกำหนดฟิลด์กิจกรรมแล้วจะต้องนำไปเขียนโปรแกรม คือที่เห็นถัดไป โดยมีชื่อสกุลกำกับชื่อกิจกรรมแยกจากกันด้วยเครื่องหมาย . (จุด) สำหรับเทอร์โบปาสคาลหรือ :: สำหรับ C++

ใช้งานสกุล

เมื่อสร้างสกุลแล้วสามารถนำมาใช้งานได้โดย ดังที่เห็นในโปรแกรมหลักคือ

1. กำหนดตัวแปร app เป็นอินสแตนซ์ของสกุล
2. ให้กำเนิดออปเจ็กต์ด้วย Init ซึ่งจะเป็นการ
   - ให้ค่าชื่อแอพพลิเคชันแก่ฟิลด์ข้อมูล
   - ในสกุลบรรพบุรุษคือ TApplication จะมีการจองหน่วยความจำ (และทำงานบางอย่าง)
   - คอมพายเลอร์จะกำหนดตารางต่าง ๆ
3. ส่งสื่อความเช่น Run ให้แก่ออปเจ็กต์ของสกุลนี้คือ app (ซึ่งขณะนี้อยู่ในฐานะออปเจ็กต์) อาจมีสื่อความอื่นได้เช่นกัน และช่วงตอนนี้ถือว่าเป็นช่วงตอนของการใช้งานสกุลที่แท้จริง
4. ยุบออปเจ็กต์ด้วย Done ซึ่งจะเป็นการยกเลิกหน่วยความจำที่จองไว้และตารางต่าง ๆ App จะหมดสภาพเป็นออปเจ็กต์ส่งสื่อความไปให้อีกไม่ได้

การสืบสกุล

การสืบสกุลหมายถึงการสืบมรดกด้วย ดังที่เห็นว่าเราสามารถใช้กิจกรรม Init, Run และ Done ซึ่งอยู่ในสกุล TApplication กับอินสแตนซ์ของสกุล TheAppC ได้ ตามหลักการ Inheri-

ลิสติงที่ 2P

```
PROGRAM CursorAndIcon;

USES WinTypes,WinProcs,WObjects;
CONST classnme = 'curico';
    appname  = 'curicon';
    thetitle = 'Cursor and Icon';

TYPE
 WndProCP = ^WndProC;
 WndProC = OBJECT(TWindow)

 constructor Init(apar : PWindowsObject; atitle : PChar);
 function  GetClassName: PChar; virtual;
 procedure GetWindowClass(var awndClass : TWndClass); virtual;
 procedure wmLButtonDown(var msg : TMessage);
       virtual wm_First + wm_LButtonDown;
 procedure DemoDisp(adc : HDC);
       END;

(* ---------- WndProC Method ---------- *)
CONSTRUCTOR WndProC.Init(apar : PWindowsObject; atitle : PChar);
BEGIN
 TWindow.Init(apar,atitle);
END;

FUNCTION WndProC.GetClassName : PChar;
BEGIN GetClassName := classnme  END;

PROCEDURE WndProC.GetWindowClass(var awndClass : TWndClass);
BEGIN
 TWindow.GetWindowClass(awndClass);
 awndClass.hIcon := LoadIcon(0, idi_Asterisk);
 awndClass.hCursor := LoadCursor(0, idc_Ibeam);
END;

PROCEDURE WndProC.wmLButtonDown(var msg : TMessage);
VAR wndDC : HDC;
BEGIN
 wndDC := GetDC(HWindow);
 DemoDisp(wndDC);
```

ลิสติงที่ 2P (ต่อ)

```
  ReleaseDC(HWindow, wndDC);
END;

PROCEDURE WndProC.DemoDisp(adc : HDC);
VAR txt1 : PChar;
BEGIN
 txt1 := 'For Windows';
 Rectangle(adc, 10,10,160,110);
 TextOut(adc, 15,15, 'TURBO Pascal',12);
 TextOut(adc, 20,35, txt1, lstrlen(txt1));
END;


(* ---------------- TheAppC Class ---------------- *)

TYPE
 TheAppC = OBJECT(TApplication)
  procedure InitMainWindow; virtual;
      END;
```

```
PROCEDURE TheAppC.InitMainWindow;
BEGIN
 MainWindow := New(WndProCP, Init(nil,thetitle));
END;

(* ---------------- MAIN PROGRAM ---------------- *)

VAR app : TheAppC;

BEGIN
 app.Init(appname);
 app.Run;
 app.Done;
END.

WVSPrintF(buf,'Error number %d',res);
```

tance (คือคอมพายเลอร์ของภาษาที่เป็นโอโอแอลจะต้องดำเนินการให้เป็นไปตามหลักการนี้)

การเปลี่ยนแปลงการทำงาน

ใน Init ของ TApplication จะเรียกใช้ InitMainWindow ของตน (ทราบจากคำอธิบายคือคู่มือสกุล) และได้กำหนดกิจกรรม InitMainWindow ไว้เป็นเวอร์ชวล (virtual) เพื่อให้เราสามารถเปลี่ยนแปลงการทำงานของกิจกรรมนี้ในผู้สืบสกุล โดย

- กำหนดกิจกรรมให้มีชื่อซ้ำกันและเป็นเวอร์ชวล (คือต้องมีส่วนหัวเหมือนของเดิมทุกประการ)
- เขียนโปรแกรมของกิจกรรมนี้ใหม่ตามที่ต้องการ

เมื่อดำเนินการตามนี้แล้วจะทำให้ Init ของสกุล TApplication เรียกใช้ InitMainWindow ของ TheAppC แทน ตามหลักการ Pohjmarphism

การต่อเติมโปรแกรม

ตามที่กล่าวแล้วว่าการปรับปรุงสกุลทำได้ 2 ลักษณะคือ การเพิ่มกับการเปลี่ยนแปลง การเพิ่มกิจกรรมไม่มีในสกุลนี้ขอให้ดูในการกำหนดสกุล WndProC ซึ่งอยู่ตอนต้น กิจกรรม DemoDisp ไม่มีในสกุล TWindow มาก่อน เพิ่มขึ้นเพื่อเพิ่มการทำงานของสกุลนี้ เป็นการต่อเติมโปรแกรม ซึ่งอยู่ในหลักการ Inheritance

คอนสตรัคเตอร์และดีสตรัคเตอร์

กิจกรรม Init และ Done เป็นคอนสตรัคเตอร์คือ constructor และดีสตรัคเตอร์คือ destructor ตามลำดับ ซึ่งในการกำหนดฟิลด์กิจกรรมจะต้องใช้คำภาษาอังกฤษทั้งสองนี้แทนโพรซีเดอร์ ความหมายคือเป็นผู้สร้างหรือให้กำเนิดและผู้ทำลายคือยุบตามลำดับ

ส่วน C++ ใช้ชื่อสกุลเป็นทั้งคอนสตรัคเตอร์และดีสตรัคเตอร์ โดยที่เมื่อหมายถึงดีสตรัคเตอร์ให้ใช้เครื่องหมาย ~ (tidle) นำหน้า และมีข้อกำหนดว่า

1. ในการสร้างสกุลต้องมีคอนสตรัคเตอร์ทุกสกุล ส่วนดีสตรัคเตอร์ให้มีเมื่อต้องการเขียนโปรแกรม หากไม่มีคอมพายเลอร์จะสร้างดีสตรัคเตอร์เพื่อทำงานในส่วนของตน
2. เมื่อนำไปใช้
   - เมื่อกำหนดอินสแตนซ์ให้ผ่านค่าไปให้เท่ากับเป็นการเรียกใช้คอนสตรัคเตอร์และเป็นกิจกรรมแรกไปใน

ตัว เป็นการให้กำเนิดออปเจ็กต์

- เมื่อจบการทำงาน ณ ที่ให้กำเนิด กรณีนี้คือใน WinMain ตรงส่งค่ากลับคือ return จะเป็นการเรียกใช้ดีสตรัคเตอร์โดยอัตโนมัติ

ด้วยวิธีนี้ทำให้ C++ มีความเป็นโอโอแอลมากกว่าเทอร์โบปาสคาล เพราะตามรูปการเขียนของเทอร์โบปาสคาล Init จะเป็นเมสเสจด้วย ซึ่งตามหลักการของโอโอพีจะส่งเมสเสจให้แก่ออปเจ็กต์ได้ก็ต่อเมื่อได้ให้กำเนิดออปเจ็กต์แล้ว

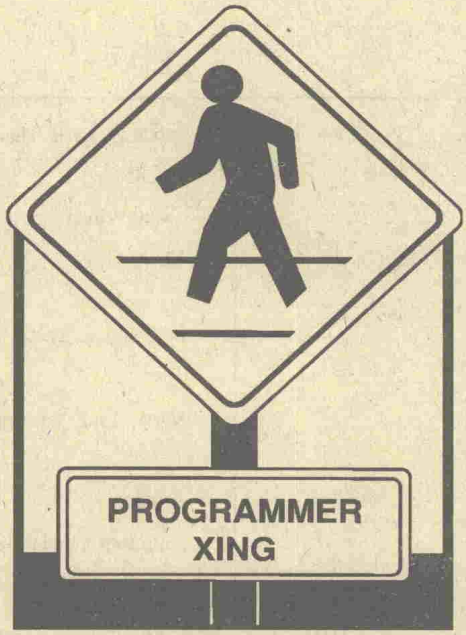
ตามที่กล่าวมานี้และเมื่อรวมกับกิจกรรมเวอร์ชวลแบบมีอินเด็กซ์ (ที่จะกล่าวต่อไป) ผมว่าเป็นความรู้ทางด้านโอโอพีที่เพียงพอที่จะเขียนโปรแกรมบนวินโดวส์แล้วครับ

สื่อความ

ผมนำคำว่าสื่อความมาใช้ก็ขอทำความเข้าใจ กล่าวคือ ในขั้นตอนของการสร้างสกุล เช่น ในโปรแกรมของกิจกรรม GetWindowClass ของสกุล WndProC เมื่อต้องการใช้กิจกรรมในสกุลบรรพบุรุษที่มีชื่อซ้ำกัน ให้ระบุชื่อสกุลบรรพบุรุษกำกับแยกจากกันด้วยจุด

แต่ในขั้นตอนของการเขียนโปรแกรมใช้งานสกุล เช่น ในโปรแกรมหลักซึ่งต้องกำหนดตัวแปรเป็นอินสแตนซ์และทำให้เป็นออปเจ็กต์แล้ว เมื่อต้องการใช้กิจกรรมให้ระบุชื่ออินสแตนซ์กำกับแยกจากกันด้วย . (จุด) ดังเช่น app.Run และถือว่าเป็นการส่งสื่อความไปให้ออปเจ็กต์

คำว่าสื่อความในภาษาอังกฤษใช้คำว่า message ผมจึงใช้เป็นสื่อความเพื่อแยกจากเมสเสจของวินโดวส์



เมสเสจ

เลยขอกล่าวคือ เมสเสจ (message) ของวินโดวส์ด้วย กล่าวคือ เมื่อผู้ใช้ดำเนินการ วินโดวส์จะส่งเมสเสจจากการดำเนินการหรือเป็นผลจากการดำเนินการให้แก่แอพพลิเคชันโดยใส่ไว้ในคิว ซึ่งแอพพลิเคชัน (โปรแกรมหลัก) จะต้องถอนจากคิวส่งให้วินโดวส์ส่งให้หน้าต่าง (คือโปรแกรมของหน้าต่าง) ตามที่กล่าวมาแล้ว

เมสเสจมีมากด้วยกัน (ประมาณ 200) โดยทำเป็นค่าคงตัวเลขจำนวนเต็มไว้ และเรียกชื่อตามชื่อค่าคงตัว เช่น เมื่อผู้ใช้กดปุ่มซ้ายของเม้าส์ วินโดวส์จะส่งเมสเสจ WM_LBUTTONDOWN ซึ่งในภาษาปาสคาลสามารถเขียนเป็นตัวเล็กใหญ่ได้ เช่น WM_LButtonDown และต่อไปผมจะเขียนตามปาสคาลเพราะอ่านได้ง่ายกว่า

เมสเสจนี้จะอยู่ในข้อมูลแบบเรคอร์ดอีกทอดหนึ่ง ซึ่งในเทอร์โบทั้งคู่จะถอดออกนำค่าเมสเสจมาทำเป็นอินเด็กซ์ เพื่อใช้กับกิจกรรมเวอร์ชวลแบบมีอินเด็กซ์ ซึ่งเมื่อเราต้องการรับเมสเสจใดให้จัดกิจกรรมสนองตอบเมสเสจนั้น ดังนี้

1. ใช้ชื่อกิจกรรมตามชื่อเมสเสจโดยไม่มีขีดล่าง และสำหรับเทอร์โบซีต้องให้ WM และอักษรที่เป็นต้นคำเป็นอักษรใหญ่ นอกนั้นเป็นอักษรเล็ก

2. ให้เป็นกิจกรรมเวอร์ชวลชนิดมีอินเด็กซ์ โดยให้ค่าอินเด็กซ์เป็น ค่าฐาน WM_FIRST บวกด้วยค่าเมสเสจ คือดังที่เห็นในโปรแกรมครับ

โปรแกรมหลัก

ตอนนี้ผมพร้อมที่จะอธิบายโปรแกรม โดยจะอธิบายทั้งสองภาษาควบคู่กันไป ข้อแตกต่างกันคือ โปรแกรมหลักของปาสคาลอยู่ใน BEGIN END. ส่วนของซีอยู่ในฟังก์ชัน main ซึ่งเมื่อเขียนโปรแกรมบนวินโดวส์ให้ใช้ชื่อเป็น WinMain รูปแบบของการเขียนโปรแกรมจึงแตกต่างกัน นอกนั้นเหมือนกัน กล่าวคือ

1. เมื่อให้กำเนิดออปเจ็กต์คือที่เรียกใช้คอนสตรัคเตอร์ จะเรียกใช้กิจกรรมต่าง ๆ เพื่อดำเนินการในเรื่องจดทะเบียนคลาสของหน้าต่าง ให้กำเนิดและให้แสดงหน้าต่าง

2. เมสเสจ Run ทำให้เข้าทำงานในวงรอบการรัน

3. เมื่อผู้ใช้ยุติการรันจะมาทำงานที่ Done หรือ return เพื่อยุบออปเจ็กต์ตามที่กล่าวมา

ค่าที่ให้ส่งกลับด้วย return เป็นค่าที่วินโดวส์ต้องการ ส่วนของเทอร์โบปาสคาล เข้าใจว่าอยู่ในสกุล TApplication

โปรแกรมของหน้าต่าง

การสร้างสกุลของหน้าต่างคือ WndProC ดำเนินการเช่นเดียวกับของโปรแกรมหลัก ซึ่งของเทอร์โบปาสคาลไม่จำเป็นต้องมี Init (เฉพาะโปรแกรมนี้) และของเทอร์โบซีไม่ต้องแยกคอนสตรัคเตอร์มาเขียนต่างหาก ที่เขียนเช่นนี้เพราะเป็นรูปแบบการเขียนโปรแกรมบนวินโดวส์โดยทั่วไป และต้องการใช้โปรแกรมนี้เป็นโปรแกรมโครงตามที่จะกล่าวต่อไป

ตามที่กล่าวว่าโปรแกรมของหน้าต่างคือโปรแกรมงาน แต่จะมีโปรแกรมส่วนอื่นผสมอยู่ด้วย แยกตามกิจกรรมได้ดังนี้

1. กิจกรรมโอโอพี

ได้แก่ คอนสตรัคเตอร์และดีสตรัคเตอร์

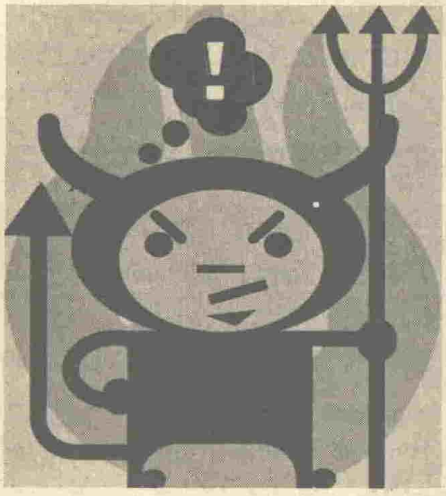
2. กิจกรรมหน้าต่าง

ดำเนินการเดียวกับหน้าต่างโดยเฉพาะคือ

- GetClassName เพื่อตั้งชื่อคลาสตามที่ได้ทำเป็นค่าคงตัวไว้ต้นโปรแกรม
- GetWindowClass เพื่อที่จะโหลดไอคอนและเคอร์เซอร์ให้ ที่เรียกใช้กิจกรรมนี้ของ TWindow เพื่อให้ได้ค่าอื่น ๆ ก่อน แล้วเปลี่ยนเฉพาะ 2 ค่านี้

ความจริงสองกิจกรรมนี้ควรอยู่ในสกุล TApplication แต่ถูกไขว้มาอยู่ใน TWindow ซึ่งในสกุล TApplication จะเรียกใช้ผ่าน MainWindow ตามที่ให้ค่าไว้ใน InitMainWindow

หากเราไม่มีสองกิจกรรมนี้ Twindow จะให้เป็นค่าที่เตรียมไว้ ซึ่งจะเหมือนกันหมด แต่ทุกแอพพลิเคชันควร



จะต้องมีชื่อคลาสและไอคอนของตนเอง

3. กิจกรรมงาน

คือกิจกรรมที่ทำงานของหน้าต่างนั้น ส่วนมากจะเป็นกิจกรรมดักรับเมสเสจของวินโดวส์ สำหรับโปรแกรมนี้คือ WMLBultonDown ซึ่งได้แก่

- ขอดีซีมาเพื่อใช้ในการแสดงผล (เรื่องของดีซีจะกล่าวต่อไป)
- ให้ DemoDisp เป็นผู้แสดงผล
- ส่งคืนดีซี

4. กิจกรรมสนับสนุน

คือโปรแกรมย่อยของกิจกรรมต่าง ๆ ซึ่งตามปกติจะไม่ใช้เป็นสื่อความเพราะเป็นกิจกรรมที่ไม่สมบูรณ์ในตัวเอง และบางกิจกรรมอาจไม่ให้ผู้สืบสกุลใช้ สำหรับของโปรแกรมนี้คือ DemoDisp ซึ่งจะวาดรูปสี่เหลี่ยมและแสดงข้อความ 2 ข้อความเพื่อจะแสดงการใช้ TextOut ด้วย การที่แยกโปรแกรมส่วนนี้มาเขียนเป็นกิจกรรม DemoDisp เพราะ

- เป็นวิธีการโดยทั่วไป เนื่องจากจะมีโปรแกรมส่วนอื่นอีกที่จะให้แสดงผล
- เพื่อแสดงการเพิ่มกิจกรรม

จากโปรแกรมจะเห็นว่าเราไม่ได้ใช้ออปเจ็กต์ของสกุลนี้โดยตรง คงกำเนิดให้หน้าต่างหลักเป็นผู้ใช้

ดีไวซ์คอนเท็กซ์

การแสดงผลของวินโดวส์ไม่ได้แสดงกับอุปกรณ์โดยตรง หากจะเอาต์พุตไปยังสิ่งที่วินโดวส์เรียกว่าดีไวซ์คอนเท็กซ์ (device context) ก่อน แล้วให้ดีไวซ์ไดรเวอร์ของอุปกรณ์นั้นนำไปแสดงผลทางอุปกรณ์ของตนต่อไป

ทำให้การเขียนโปรแกรมไม่ขึ้นกับอุปกรณ์ เช่น เมื่อเป็นจอภาพก็ไม่ต้องคำนึงว่าเป็นจอภาพชนิดใด เรียกว่าเป็น GDI (Graphics Device Independence)

สำหรับค่าดีไวซ์คอนเท็กซ์ ซึ่งนิยมเรียกว่าค่าดีซี จะต้องขอสำหรับอุปกรณ์แต่ละชนิด โดยที่ของจอภาพให้ขอด้วย GetDC จะได้เป็นดีซีของหน้าต่างบานที่เราผ่านค่าแฮนเดิลของหน้าต่างคือ HWindow ไปให้

เมื่อใช้งานเสร็จแล้วจะต้องส่งคืนด้วย ReleaseDC เพราะมีจำกัด (5 หน่วย) หากใช้จนหมดผู้ใช้จะดำเนินการใดไม่ได้เลยแม้แต่จะยุติการทำงานกับวินโดวส์

ค่าแฮนเดิลของหน้าต่างหรือของสิ่งอื่นใดก็ตาม คือค่าแฮนเดิลที่วินโดวส์จัดให้เมื่อเราให้กำเนิดสิ่งนั้น ๆ สำหรับของหน้าต่างเทอร์โบทั้งคู่จะเก็บไว้ในฟิลด์ HWindow ของหน้าต่างบานนั้น เมื่อเราเขียนโปรแกรมให้หน้าต่างบานใดจึงสามารถนำมาใช้ได้โดยตรง

ค่าแฮนเดิลคืออะไรเรื่องยาวครับ ขอให้ถือว่าเป็นค่าที่จะใช้อ้างอิงสิ่งต่าง ๆ (ที่มีค่าแฮนเดิล) และเป็นค่าเฉพาะเช่นของหน้าต่างแต่ละบาน เป็นต้น

โครงโปรแกรม

ถึงแม้ในเทอร์โบทั้งคู่จะนำส่วน

ของโปรแกรมที่ต้องเขียนซ้ำในทุกแอพพลิเคชันมาทำเป็นสกุล แต่เมื่อเรานำมาเขียนโปรแกรมแบบโอโอพีก็ยังมีส่วนที่ซ้ำกันในทุกแอพพลิเคชันอีก กล่าวคือ

ในส่วนของสกุล TheAppC และตัวโปรแกรมหลัก ส่วนที่แตกต่างกันคือชื่อโปรแกรมในภาษาปาสคาล ซึ่งในภาษาซีได้ให้ทำเป็นหมายเหตุไว้ด้วย และที่ทำเป็นค่าคงตัวไว้คือ

- classname เป็นชื่อคลาสของหน้าต่างหลัก ควรใช้ชื่อเดียวกับชื่อไฟล์
- appname เป็นชื่อแอพพลิเคชันอันเป็นส่วนของเทอร์โบ ควรใช้ชื่อย่อของชื่อโปรแกรม
- Thetitle คือข้อความที่ให้แสดงเป็นไตเติล

นั่นคือจะเปลี่ยนแปลงเฉพาะตรงนี้เท่านั้น (ผมจึงได้ทำเป็นค่าคงตัว) ขอให้ดำเนินการดังนี้ครับ ก๊อบปี้ไฟล์ CURICO.PAS หรือ CURICO.CPP เป็นไฟล์ 1A.PAS หรือ 1A.CPP แล้วดำเนินการในส่วนของ WndProC ดังนี้คือ

1. ลบกิจกรรม DemoDisp (ลบทั้งฟิลด์กิจกรรมและโปรแกรมของกิจกรรม)

2. ลบโปรแกรมใน WMLButtonDown รวมทั้งการกำหนดตัวแปร WndDC ด้วย คงเหลือเป็น BEGIN END; หรือ { }

3. ในโปรแกรมของกิจกรรม GetWindowClass ให้ทำบรรทัดที่ LoadCursor เป็นหมายเหตุ หรือเปลี่ยนค่า ide_Ibeam เป็น ide_Arraw (ภาษาซีเป็นตัวใหญ่ทั้งหมด) จะให้ผลเท่ากัน

เก็บลงไฟล์ไว้ใช้เป็นโครงโปรแกรมต่อไป ซึ่งเมื่อดำเนินการเช่นนี้นอกจากประหยัดเวลาป้อนโปรแกรมของคุณแล้ว ผมก็ไม่ต้องแสดงโปรแกรมทั้งหมดด้วย ทำให้สามารถแสดงโปรแกรมเบ็ดเตล็ดได้มากขึ้น

ใช้โครงโปรแกรม

เราจะเริ่มใช้โครงโปรแกรมเลยครับ สมมติว่าผมได้อธิบายความบางเรื่องแล้วและกล่าวว่า "ขอให้ป้อนโปรแกรม Painter1 ตามลิสติงที่ 3p หรือ 3c เก็บไว้ในไฟล์ PAINT1.PAS หรือ PAINT1.CPP โดยใช้โครงโปรแกรม"

ความหมายคือ จากโครงโปรแกรมจะต้องเพิ่มฟิลด์ข้อมูล 2 ฟิลด์และฟิลด์กิจกรรม 2 ฟิลด์จะดำเนินการได้ดังนี้

1. ลอกไฟล์ 1A.PAS หรือ 1A.CPP เป็น PAINT1.PAS หรือ PAINT1.CPP แล้วเปิดไฟล์ด้วยชื่อ

ลิสติงที่ 3P

```
PROGRAM Painter1;

USES  WinTypes,WinProcs,WObjects;
CONST classnme = 'paint1';
      appname  = 'painter1';
      thetitle = 'Painter';


TYPE
 WndProCP = ^WndProC;
 WndProC = OBJECT(TWindow)
  wndDC  : HDC;
  pendwn : Boolean;
  .....
 procedure  wmMouseMove(var msg : TMessage);
         virtual wm_First + wm_MouseMove;
 procedure  wmLButtonUp(var msg : TMessage);
         virtual wm_First + wm_LButtonUp;
         END;

(* ---------- WndProC  routine ---------- *)

CONSTRUCTOR  WndProC.Init(apar : PWindowsObject; atitle : PChar);
BEGIN
 TWindow.Init(apar,atitle);  pendwn := false;
END;


PROCEDURE  WndProC.wmLButtonDown(var msg : TMessage);
BEGIN
 if not pendwn then
  begin
  pendwn := TRUE;
  SetCapture(HWindow);
  SetCursor(LoadCursor(0, IDC_CROSS));
  wndDC := GetDC(HWindow);
```

ลิสติงที่ 3P (ต่อ)

```
  MoveTo(wndDC, msg.lParamLo, msg.lParamHi);
  end;
END;

PROCEDURE WndProC.wmMouseMove(var msg : TMessage);
BEGIN
 if pendwn then
  LineTo(wndDC, msg.lParamLo, msg.lParamHi);
END;

PROCEDURE WndProC.wmLButtonUp(var msg : TMessage);
BEGIN
 if pendwn then
  begin
  pendwn := FALSE;
  ReleaseCapture;
  ReleaseDC(HWindow, wndDC);
  end;
END;
```

ลิสติงที่ 3C

```
// Painter1

#include "owl.h"

#define classnme "paint1"
#define appname "painter1"
#define thetitle "Painter"

_CLASSDEF (WndProC)
class WndProC : public TWindow {
public:
 HDC wndDC;
 BOOL pendwn;
 .....
 virtual void WMMouseMove(RTMessage msg) =
   [WM_FIRST + WM_MOUSEMOVE];
 virtual void WMLButtonUp(RTMessage) =
   [WM_FIRST + WM_LBUTTONUP];
};


/* ---------------------- WndProC routine ---------------- */

WndProC::WndProC(PTWindowsObject AParent, LPSTR ATitle)
 : TWindow(AParent, ATitle)
{
 pendwn = FALSE;
}

void WndProC::WMLButtonDown(RTMessage msg)
{
 if ( !pendwn )
 {
  pendwn = TRUE;
  SetCapture(HWindow);
  SetCursor(LoadCursor(0, IDC_CROSS));
  wndDC = GetDC(HWindow);
  MoveTo(wndDC, msg.LP.Lo, msg.LP.Hi);
 }
}

void WndProC::WMMouseMove(RTMessage msg)
{
 if (pendwn)

  LineTo(wndDC, msg.LP.Lo, msg LP.Hi);
}


void WndProC::WMLButtonUp(RTMessage)
{
 if (pendwn)
 {
  pendwn = FALSE;
  ReleaseCapture();
  ReleaseDC(HWindow, wndDC);
 }
}
```

2. เปลี่ยนชื่อโปรแกรมและค่าที่เกี่ยวข้องให้เป็นดังในลิสติงใหม่นี้

3. เพิ่มฟิลด์ข้อมูล wndDC กับ pendwn

4. ป้อนฟิลด์กิจกรรมซึ่งอาจจะดำเนินการได้โดย

- ลอก (Edit/Copy) ฟิลด์กิจกรรม wmLBultonDown ทั้งฟิลด์แล้วปะ (Edit/Paste) ไว้ข้างใต้

- แก้คำว่า Down เป็น Up ทั้งที่ชื่อกิจกรรมและชื่อเมสเสจ

- ลอกปะอีกครั้งแล้วเปลี่ยนคำว่า LBultonDown เป็น MouseMove ทั้งสองแห่ง

5. ป้อนโปรแกรมในคอนสตรัคเตอร์คือที่ให้ pendwn เป็นเท็จ

6. ก๊อบปี้โปรแกรมของกิจกรรม wmLButtonDown เป็น wmLBultonWP และ wmMouseMove แล้วป้อนโปรแกรมตามลิสติง

เท่านี้ครับจะได้โปรแกรมที่พร้อมจะคอมพายล์และรัน และเมื่อรันโปรแกรมนี้แล้วคุณสามารถคลิ๊กเม้าส์ซ้ายแล้วเลื่อนไปมาวาดรูปได้ จะเห็นว่าการเขียนโปรแกรมบนวินโดวส์ไม่ยากใช่ไหมครับ สำหรับคำอธิบายโปรแกรมจะกล่าวในฉบับหน้า ขอให้ลองคิดดูว่าเป็นไปได้อย่างไร

ซ้อมมือ

หากคุณสนใจที่จะเขียนโปรแกรมบนวินโดวส์จริงจัง ขอให้ป้อน คอมพายล์ และรันโปรแกรมทั้งสามนี้ด้วยตัวคุณเอง แล้วควรทดลองเขียนโปรแกรมเพิ่มเช่น

1. ในโปรแกรม CursorAndIcon ขอให้ทดลอง

- แสดงข้อความต่าง ๆ ณ ตำแหน่งต่าง ๆ ในหน้าต่าง

รูปที่ 5
คำสั่งที่จะใช้ในการทดลองเขียนโปรแกรม

```
pnt1,pnt2 : TPoint;
pnt1 := MakePoint(msg.lParam);
pnt2 := MakePoint(msg.lParam);
Rectangle(wndDC, pnt1.x, pnt1.y, pnt2.x, pnt2.y);
```

```
POINT pnt1, pnt2;
pnt1 = MAKEPOINT(msg.LParam);
pnt = MAKEPOINT(msg.LParam);
Rectangle(wndDC, pnt1.x, pnt1.y, pnt2.x, pnt2.y);
```

- แสดงรูปสี่เหลี่ยมขนาดต่าง ๆ ณ ตำแหน่งต่าง ๆ และรูปวงรี ใช้ฟังก์ชัน Ellipse (การวาดภาพด้วยวิธีนี้จะเป็นการวาดทับ)

2. ในโปรแกรม Painter1 อาจจะมากกว่าแต่ขอให้ทดลองครับ เช่น เปลี่ยนเป็นให้ผู้ใช้วาดภาพสี่เหลี่ยมแทน โดยที่เมื่อผู้ใช้จะวาดรูปสี่เหลี่ยมให้คลิ๊กเม้าส์ซ้าย ณ ตำแหน่งที่จะให้เป็นจุดมุมบนขวาคือ จุดแรก แล้วเลื่อนเม้าส์ไปปล่อยมือ ณ ตำแหน่งที่จะให้เป็นจุดมุมล่างขวาคือสุดหลัง

จึงต้องเก็บจุดแรกไว้เป็นฟิลด์ข้อมูลฟิลด์หนึ่งให้เป็นข้อมูลแบบ TPoint และโดยที่วินโดวส์จะให้ตำแหน่งของเคอร์เซอร์มากับ lParam ในเมสเสจทั้งสามเป็น LongInt สามารถใช้ MakePoint แปลงเป็น TPoint ได้ ได้แสดงคำสั่งที่จะใช้ไว้ในรูปที่ 5 โดยให้จุดแรกแก่ pnt1

สำหรับจุดหลังไม่จำเป็นต้องเก็บค่าไว้ แต่ผมให้เก็บไว้ใน pnt2 และสามารถแสดงรูปสี่เหลี่ยมได้ด้วย Rectangle ในกิจกรรม wmLBultonUp ขอให้ทดลองเขียนโปรแกรมโดยนำคำสั่งเหล่านี้ไปแทรกในที่เหมาะสมครับ

ท้ายตอน

เป็นอันว่าในตอนนี้คุณได้โปรแกรมถึง 2 โปรแกรมไว้ซ้อมมือ และได้ทราบเรื่องของการเขียนโปรแกรมบนวินโดวส์หลายเรื่องเช่นกัน คือ

1. การเขียนโปรแกรมแบบโอโอพี (ที่น่าจะเพียงพอ) สำหรับการเขียนโปรแกรมบนวินโดวส์

2. จุดเริ่มต้นของการเขียนโปรแกรมบนวินโดวส์ (ตามที่ทำเป็นโครงโปรแกรมไว้)

3. ที่เกี่ยวข้องกับการเขียนโปรแกรมบนวินโดวส์แท้ ๆ คือ

- การให้กำเนิดหน้าต่างหลัก
- การเขียนโปรแกรมดักรับเมสเสจ
- การเขียนโปรแกรมจากเหตุ (ซึ่งยังไม่ชัดเจนนักจะทำให้ชัดเจนในตอนต่อ ๆ ไป)
- การแสดงผลเป็นภาพและข้อความ (เบื้องต้น)
- การเขียนโปรแกรมให้ใช้เม้าส์วาดรูป

หากคุณได้ดำเนินการตามที่กล่าวในหัวข้อ "ซ้อมมือ" ก็นับว่าได้ตั้งไข่พร้อมที่จะเดินและวิ่งในฉบับต่อ ๆ ไปครับ